EMULEX XENIX DRIVER EXAMPLE PACKAGE

USER'S GUIDE

3545 Harbor Boulevard
Costa Mesa, California 92626
(714) 662-5600 TWX 910-595-2521

VD9950903 Rev A
August 1985

The information in this manual is for information purposes and is subject to change without notice.

Emulex Corporation assumes no responsibility for any errors that may appear in the manual.

Printed in U.S.A.

TABLE OF CONTENTS

BLANK

Section 1
GENERAL DESCRIPTION

1.1 INTRODUCTION

This manual explains the purpose and use of the Emulex Xenix Driver Example Package, which includes the following items:

- Source text files on diskette
- Source listing with explanatory text

1.2 PRODUCT OVERVIEW

In order to use the Emulex MB01 Multibus Host Adapter in a Xenix operating system environment, you must first install a software driver. The Emulex Xenix Driver presented in Section 2 of this document is an **example** of such a driver.

This sample driver has been tested with the following system configuration:

- Intel 310 CPU
- Xenix operating system
- Emulex MD01 (Medalist) SCSI disk controller
- Emulex MT01 (Titleist) SCSI tape controller
- Emulex MB01 Multibus host adapter

Before using the driver, you must modify it as necessary to suit your system configuration and then integrate it into your operating system.

NOTE

Emulex does not support this driver, because it is not intended for use in all Multibus environments. Each system may have slightly different driver requirements that are not met by this example. This document assumes that the user has the expertise required to customize the driver.

1.3 DISTRIBUTION MEDIA

As shown in Table 1-1, the driver is contained in two files on the distribution diskette: HA.H and HA.C. HA.H contains the definitions used by the driver, and HA.C contains the sample driver itself, with a source statement to include file HA.H.

Table 1-1. Distribution Media

Emulex P/N	Description	Contents
VD9960704	Xenix floppy diskette	HA.H HA.C

To retrieve these two files from the distribution diskette, use the Xenix **tar** command with the **f** option to specify the special file name of the drive from which the diskette is to be read, and the **x** option to specify that files HA.C and HA.X are to be extracted. If the **v** (verbose) option is used, the screen will display a message stating that files HA.C and HA.X are being extracted. An example of this command follows:

tar xfv /dev/rdf0<return>

1.4 COMPATIBILITY AND REQUIREMENTS

1.4.1 HARDWARE

The Emulex Xenix Driver Example has been tested and found to be compatible with the Emulex MB01 Multibus Host Adapter, MD01 (Medalist) Disk Controller, MT01 (Titleist) Tape Controller, and Intel 310 CPU. The user must modify the driver for use with other hardware configurations.

1.4.2 SOFTWARE

The Emulex Xenix Driver Example is adaptable for use with the Intel Xenix operating system.

1.5 RELATED DOCUMENTATION

Title: MB01 Multibus Host Adapter Technical Manual
Publication Number: HA5151001
Publisher: Emulex Corporation
3545 Harbor Blvd.
Costa Mesa, CA 92626
(714) 662-5600 TWX 910-595-2521

2.1 OVERVIEW

This section contains the source listing for the Xenix driver example provided with this package. It was written for and tested in a specific configuration, and must be adapted to your specific requirements.

2.2 SOURCE LISTING

Figure 2-1 contains the source listing.

```
typedef char            UCHAR;

#define ADDR_CONV       0x4000L /* address conversion                */

#define B_IOCTL         0x4000  /* ioctl I/O flag for buf header*/

#define SET_LOAD_BIT    1       /* tape load/unload command          */
#define RESET_LOAD_BIT  0       /* tape load/unload command          */

#define TF_WRITTEN      1       /* wrote to tape                     */
#define TAPE            0x80    /* mode tape flag                    */

#define TAPE_IS_PRESENT TRUE    /* tape present constant             */
#define TAPE_NOT_PRESENT FALSE  /* tape not present constant         */

#define TMO_INT         360     /* interrupt timeout                 */
#define TMO_RES         300     /* reselection timeout               */

#define TIMER_RESET     0       /* timer is not set                  */
#define TIMER_SET       1       /* timer is set                      */
#define TIMER_RUNNNG    1       /* timer is active                   */
#define TIMER_NOT_RUNNING 0     /* timer is inactive                 */

#define ARM_TIMER(x)    ha_tinfo.ha_tmo = (x)
#define INIT_DELAY      50      /* initialize time delay             */

#define BIT_UNIT        0x80    /* messase in unit flag bit          */

#define MAX_SB          0x1fffff /* 6 byte iocb block max value */
#define MAX_IOCB_SZ     5       /* words max in iocb                 */
#define MAX_BYTES_IOCB  MAX_IOCB_SZ * 2 /* max iocb bytes            */
#define MAX_SCSI_UNITS  8       /* number of SCSI units on bus       */
#define MAX_LUN         8       /* max LUNs per SCSI device          */
#define MAX_SZ          20      /* max buffer union elements         */
#define MAX_WAIT_TIME   0xfff   /* maximum timeout value             */
#define MAX_ERRORS      2       /* maximum I/O errors allowed        */
```

Figure 2-1. Source Listing for Xenix Driver Example
(Continued on next page)

```
/* Definitions for accessing vars as chars, short, or int          */

#define SZ_CHARS        (MAX_SZ)        /* number of chars          */
#define SZ_SHORTS       (MAX_SZ/2)      /* number of shorts         */
#define SZ_INTS         (MAX_SZ/4)      /* number of ints           */
#define SZ_REPLY        12              /* size of reply            */

#define MASK_NIB        0x0f    /* nibble mask                      */
#define MASK_BYTE       0xff    /* byte mask                        */
#define MASK_WORD       0xffff  /* word mask                        */
#define MASK_DEBUG      7       /* isolate value for ha_debug       */
#define MASK_UNIT       7       /* mask for unit numbers            */
#define MASK_PHASES     7       /* mask to isolate phase bits       */
#define MASK_CNTRL      1       /* isolate controller bits          */

#define SHIFT_LUN       5       /* LUN bit shifts                   */
#define SHIFT_PHASES    3       /* phase right-aligned              */
#define SHIFT_BYTE      8       /* num of bits for byte shift       */
#define SHIFT_WORD      16      /* num of bits for word shift       */
#define SHIFT_SCSI      6       /* for controller number            */
#define SHIFT_UNIT      3       /* for unit in bp->b_dev            */

#define SWAP_INT(x)     (((x >> SHIFT_WORD) & MASK_WORD) | \
                         ((x << SHIFT_WORD) & MASK_WORD))
#define SWAP_SHORT(x)   ((((x & MASK_WORD) << SHIFT_BYTE)) | \
                         (((x & MASK_WORD) >> SHIFT_BYTE)))
#ifdef  SUN
#define SWAP_BINT(x)    (((SWAP_SHORT(x & MASK_WORD)) & MASK_WORD)  | \
                                ((SWAP_SHORT((x>>SHIFT_WORD) & 0xffff)) \
                                        <<SHIFT_WORD))
#else
#define SWAP_BINT(x)    ((SWAP_SHORT(x) & 0xffffL) | \
                         (((SWAP_SHORT(x>>SHIFT_WORD) & 0xffffL) \
                                <<SHIFT_WORD) & 0xffff0000L))
#endif

#define SEM_FREE        0       /* semaphore available              */
#define SEM_BUSY        1       /* semaphore allocated              */

#define PH_OUT_DATA     0       /* data out SCSI bus phase          */
#define PH_IN_DATA      1       /* data in phase                    */
#define PH_COMMAND      2       /* command phase                    */
#define PH_STATUS       3       /* status phase                     */
#define PH_BAD1         4       /* non-existent phase               */
#define PH_BAD2         5       /* non-existent phase               */
#define PH_OUT_MSG      6       /* message out phase                */
#define PH_IN_MSG       7       /* message in phase                 */

#define GET_STATE_FLAG  (bp->b_cylin >> SHIFT_BYTE)
#define SET_STATE_FLAG(x) bp->b_cylin = bp->b_cylin | ((x) << SHIFT_BYTE)

/* Phase States                                                     */

#define SF_SEL          0x01    /* selection state                  */
#define SF_DCN          0x02    /* disconnected                     */
#define SF_CMP          0x04    /* command complete state           */
#define SF_MSGI         0x08    /* message in phase                 */
#define SF_ABT          0x10    /* abort state                      */
#define SF_IO           0x20    /* I/O direction state              */
#define SF_DAT          0x40    /* data phase state                 */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#define scsilun(x)      ((x >> SHIFT_UNIT) & MASK_UNIT)
#define scsidev(x)      ((x >> SHIFT_SCSI) & MASK_CNTLR)
#define tscsidev(x)     ((x) & MASK_UNIT)

#define TRUE            1       /* define truth value           */
#define FALSE           0       /* define truth value           */

#define SPL()           spl5()  /* ha interrupt priority level  */
#define INT_NORM_LEVEL  2       /* interrupt level              */
#define INT_TMO_LEVEL   8       /* psuedo interrupt level       */

#define FS              5       /* number of disk file systems  */
#define FS_ROOT         0
#define FS_SWAP         1
#define FS_USR          2
#define FS_FAST         3
#define FS_BOOT         4

#define DISK_UNITS      1       /* number of disks per cntrl    */
#define TAPE_UNITS      1       /* number of tapes per cntrl    */
#define CNTLR_UNITS     2       /* number of disk controllers   */
#define HA_UNITS        2       /* total controllers            */

#define S512            512     /* sector size of 512 bytes     */
#define SZ512           2       /* sector divide factor         */
#define SZ1024          1       /* sector divide factor         */
#define ID_DISK         1       /* disk controller              */
#define ID_TAPE         2       /* tape controller              */

#define SZ_SMALL_DK     0x989680L       /* 10Mb                 */
#define SZ_MEDIUM_DK    0x1312d00L      /* 20Mb                 */
#define SZ_LARGE_DK     0x2625a00L      /* 40Mb                 */
#define DK_SMALL        0       /* small capacity disk          */
#define DK_MEDIUM       1       /* medium capacity disk         */
#define DK_LARGE        2       /* large capacity disk          */

#define S_BLKS_ROOT     12000   /* root file system disk blocks */
#define S_BLKS_SWAP      6000   /* swap file number of blocks   */
#define S_BLKS_USR        100   /* usr file system disk blocks  */
#define S_BLKS_FAST       100   /* disk/tape disk buffer blocks */
#define S_BLKS_BOOT        35   /* bootstrap blocks             */

#define M_BLKS_ROOT     16000   /* root file system disk blocks */
#define M_BLKS_SWAP      8000   /* swap file number of blocks   */
#define M_BLKS_USR        300   /* usr file system disk blocks  */
#define M_BLKS_FAST       300   /* disk/tape disk buffer blocks */
#define M_BLKS_BOOT        35   /* bootstrap blocks             */

#define L_BLKS_ROOT     20000   /* root file system disk blocks */
#define L_BLKS_SWAP     10000   /* swap file number of blocks   */
#define L_BLKS_USR        600   /* usr file system disk blocks  */
#define L_BLKS_FAST       600   /* disk/tape disk buffer blocks */
#define L_BLKS_BOOT        35   /* bootstrap blocks             */


#define INIT_NO         0       /* configuration not established*/
#define INIT_YES        1       /* establishing configuration   */
#define INIT_DONE       2       /* configuration established    */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#define UNIT_OPENED     TRUE    /* unit has been opened ok      */
#define UNIT_CLOSED     FALSE   /* unit is not opened           */

#define BASE_IO_ADDR    0x1000  /* I/O space HA base address    */

#define DMA_SBT         0x40    /* DMA single byte transfer     */

#ifdef  SUN
#define DMA_ADR         BASE_IO_ADDR + 0x01     /* DMA address  */
#define DMA_CNT         BASE_IO_ADDR + 0x00     /* word count   */
#define DMA_CMD         BASE_IO_ADDR + 0x09     /* command      */
#define DMA_STS         BASE_IO_ADDR + 0x09     /* status       */
#define DMA_REQ         BASE_IO_ADDR + 0x08     /* request      */
#define DMA_MASK        BASE_IO_ADDR + 0x0b     /* mask         */
#define DMA_MODE        BASE_IO_ADDR + 0x0a     /* mode         */
#define DMA_CBPF        BASE_IO_ADDR + 0x0d     /* clear byte   */
#define DMA_TEMP        BASE_IO_ADDR + 0x0c     /* temporary    */
#define DMA_MC          BASE_IO_ADDR + 0x0c     /* master clear */
#define DMA_AMSK        BASE_IO_ADDR + 0x0e     /* all mask     */
#else
#define DMA_ADR         BASE_IO_ADDR + 0x00     /* DMA address  */
#define DMA_CNT         BASE_IO_ADDR + 0x01     /* word count   */
#define DMA_CMD         BASE_IO_ADDR + 0x08     /* command      */
#define DMA_STS         BASE_IO_ADDR + 0x08     /* status       */
#define DMA_REQ         BASE_IO_ADDR + 0x09     /* request      */
#define DMA_MASK        BASE_IO_ADDR + 0x0a     /* mask         */
#define DMA_MODE        BASE_IO_ADDR + 0x0b     /* mode         */
#define DMA_CBPF        BASE_IO_ADDR + 0x0c     /* clear byte   */
#define DMA_TEMP        BASE_IO_ADDR + 0x0d     /* temporary    */
#define DMA_MC          BASE_IO_ADDR + 0x0d     /* master clear */
#define DMA_AMSK        BASE_IO_ADDR + 0x0f     /* all mask     */
#endif

#ifdef  SUN
#define DMA_PR          BASE_IO_ADDR + 0x22     /* DMA page reg */
#else
#define DMA_PR          BASE_IO_ADDR + 0x23     /* DMA page reg */
#endif

#define DMA_IN          4       /* DMA mode - data in           */
#define DMA_OUT         8       /* DMA mode - data out          */
#define DMA_CHNL_MASK   4       /* channel mask bit             */


#define D_P1            0x20    /* set extended write/normal timing */

/* NCR 5385  SCSI Controller Registers */

#ifdef  SUN
#define DAR             BASE_IO_ADDR + 0x11     /* data register*/
#else
#define DAR             BASE_IO_ADDR + 0x10     /* data register*/
#endif

#ifdef  SUN
#define CMR             BASE_IO_ADDR + 0x10     /* cmd reg      */
#else
#define CMR             BASE_IO_ADDR + 0x11     /* cmd reg      */
#endif
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#define CMR_BYT         0x40    /* single byte transfer          */
#define CMR_DMA         0x80    /* DMA mode                      */

#ifdef  SUN
#define CTR             BASE_IO_ADDR + 0x13     /* contrl reg    */
#else
#define CTR             BASE_IO_ADDR + 0x12     /* contrl reg    */
#endif

#define CTR_SEL         0x01    /* select enable                 */
#define CTR_RES         0x02    /* reselect enable               */
#define CTR_PAR         0x04    /* parity enable                 */

#ifdef  SUN
#define DIR             BASE_IO_ADDR + 0x12     /* dest ID reg   */
#else
#define DIR             BASE_IO_ADDR + 0x13     /* dest ID reg   */
#endif

#ifdef  SUN
#define ASR             BASE_IO_ADDR + 0x15     /* aux stat reg */
#else
#define ASR             BASE_IO_ADDR + 0x14     /* aux stat reg */
#endif

#define ASR_TCZ         0x02    /* transfer count zero           */
#define ASR_PAU         0x04    /* paused                        */
#define ASR_IO          0x08    /* SCSI I/O signal               */
#define ASR_CD          0x10    /* SCSI c/d signal               */
#define ASR_MSG         0x20    /* SCSI msg signal               */
#define ASR_PE          0x40    /* parity error                  */
#define ASR_DAT         0x80    /* Data register full            */

#ifdef  SUN
#define IDR             BASE_IO_ADDR + 0x14     /* ID register   */
#else
#define IDR             BASE_IO_ADDR + 0x15     /* ID register   */
#endif

#ifdef  SUN
#define INR             BASE_IO_ADDR + 0x17     /* intr reg      */
#else
#define INR             BASE_IO_ADDR + 0x16     /* intr reg      */
#endif

#define INR_FC          0x01    /* function complete             */
#define INR_BS          0x02    /* bus service                   */
#define INR_DCN         0x04    /* disconnected                  */
#define INR_SEL         0x08    /* selected                      */
#define INR_RES         0x10    /* reselected                    */
#define INR_IVC         0x40    /* invalid command               */
#define INR_RESET       0x80    /* reset intr - driver generated*/

#ifdef  SUN
#define SIR             BASE_IO_ADDR + 0x16     /* source ID reg*/
#else
#define SIR             BASE_IO_ADDR + 0x17     /* source ID reg*/
#endif

#define SIR_IDV         0x80    /* ID valid                      */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#ifdef  SUN
#define DSR             BASE_IO_ADDR + 0x18     /* diag stat reg*/
#else
#define DSR             BASE_IO_ADDR + 0x19     /* diag stat reg*/
#endif

#define DSR_SDS         0x07    /* self diagnostic status       */
#define DSR_DCS         0x38    /* diagnostic command ststus    */
#define DSR_DC          0x80    /* diagnostic complete          */

#ifdef  SUN
#define TCR_B3          BASE_IO_ADDR + 0x1d     /* trans ctr MSB*/
#else
#define TCR_B3          BASE_IO_ADDR + 0x1c     /* trans ctr MSB*/
#endif

#ifdef  SUN
#define TCR_B2          BASE_IO_ADDR + 0x1c     /* trans counter*/
#else
#define TCR_B2          BASE_IO_ADDR + 0x1d     /* trans counter*/
#endif

#ifdef  SUN
#define TCR_B1          BASE_IO_ADDR + 0x1f     /* trans ctr LSB*/
#else
#define TCR_B1          BASE_IO_ADDR + 0x1e     /* trans ctr LSB*/
#endif

/* Other Host Adapter Registers */

#ifdef  SUN
#define GSR             BASE_IO_ADDR + 0x25     /* stat register*/
#else
#define GSR             BASE_IO_ADDR + 0x24     /* stat register*/
#endif

#define GSR_RST         0x02    /* SCSI bus reset flag          */
#define GSR_REQ         0x04    /* SCSI request signal          */
#define GSR_INT         0x38    /* interrupt level select       */
#define GDR_OPT1        0x40    /* option switch                */
#define GDR_OPT2        0x80    /* option switch                */

#ifdef  SUN
#define GCR             BASE_IO_ADDR + 0x21     /* command reg  */
#else
#define GCR             BASE_IO_ADDR + 0x20     /* command reg  */
#endif

#define GCR_RST         0x01    /* reset SCSI bus               */

#ifdef  SUN
#define RSRR            BASE_IO_ADDR + 0x26     /* release reset*/
#else
#define RSRR            BASE_IO_ADDR + 0x27     /* release reset*/
#endif

/* NCR 5385 SCSI Controller command codes */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#define AC_RST          0x00    /* chip select                      */
#define AC_DCN          0x01    /* disconnect                       */
#define AC_PAU          0x02    /* pause                            */
#define AC_ATN          0x03    /* set attention                    */
#define AC_MA           0x04    /* message accepted                 */
#define AC_CD           0x05    /* chip disabled                    */
#define AC_SWA          0x08    /* select with attention            */
#define AC_SEL          0x09    /* select without attention         */
#define AC_RES          0x0a    /* reselect                         */
#define AC_DDT          0x0b    /* diagnostic data turnaround       */
#define AC_RC           0x0c    /* receive command                  */
#define AC_RD           0x0d    /* receive data                     */
#define AC_RM           0x0e    /* receive message out              */
#define AC_RI           0x0f    /* receive unspecified info out     */
#define AC_SS           0x10    /* send status                      */
#define AC_SD           0x11    /* send data                        */
#define AC_SM           0x12    /* send message in                  */
#define AC_SI           0x13    /* send unspecified info in         */
#define AC_TI           0x14    /* transfer info                    */
#define AC_TP           0x15    /* transfer pad                     */

/* Class 0 Command Definitions  */

#define OC_READY        0x00    /* test unit ready                  */
#define OC_REZERO       0x01    /* disk rezero unit                 */
#define OC_REWIND       0x01    /* tape - rewind                    */
#define OC_SENSE        0x03    /* request sense                    */
#define OC_FORMAT       0x04    /* format unit                      */
#define OC_REASSIGN     0x07    /* reassign block                   */
#define OC_READ         0x08    /* read                             */
#define OC_WRITE        0x0a    /* write                            */
#define OC_SEEK         0x0b    /* seek                             */
#define OC_FM_WRITE     0x10    /* tape - write file mark           */
#define OC_INQUIRY      0x12    /* inquiry                          */
#define OC_SELECT_MODE  0x15    /* mode select                      */
#define OC_RESERVE      0x16    /* reserve unit                     */
#define OC_RELEASE      0x17    /* release unit                     */
#define OC_COPY         0x18    /* copy                             */
#define OC_ERASE        0x19    /* erase                            */
#define OC_MODE_SENSE   0x1a    /* mode sense                       */
#define OC_LOAD         0x1b    /* tape load/unload                 */
#define OC_RECV_DIAGS   0x1c    /* receive diagnostic               */
#define OC_SEND_DIAG    0x1d    /* send diagnostic                  */

#define OC_0DEBUG       0xf0    /* turn off ha_debug                */
#define OC_1DEBUG       0xf1    /* set ha_debug to 1                */
#define OC_2DEBUG       0xf2    /* set ha_debug to 2                */

/* Class 1 Command Definitions  */

#define CMD_GRP_1       0x20    /* group 1 command flag             */

#define OC_CAPACITY     0x25    /* read capacity                    */
#define OC_XREAD        0x28    /* extended read                    */
#define OC_XWRITE       0x2a    /* extended write                   */
#define OC_XSEEK        0x2b    /* extended seek                    */
#define OC_VERIFY       0x2f    /* verify                           */

/* SCSI Completion Status Bit Definitions */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#define CS_PE           0x01    /* parity error                    */
#define CS_ERR          0x02    /* error condition                 */
#define CS_BSY          0x08    /* target busy                     */
#define CS_ICS          0x10    /* intermediate completion stat */
#define CS_REC          0x20    /* recoverable error               */

/* SCSI Message Byte Definitions */

#define MSG_CMP         0x00    /* command complete                */
#define MSG_SDP         0x02    /* save data pointer               */
#define MSG_DCN         0x04    /* disconnect                      */
#define MSG_ABT         0x06    /* abort                           */
#define MSG_REJ         0x07    /* message reject                  */
#define MSG_NOP         0x08    /* no operation                    */
#define MSG_MPE         0x09    /* message parity error            */
#define MSG_LCC         0x0a    /* linked command complete         */
#define MSG_LCF         0x0b    /* linked cmd complete with flas*/
#define MSG_RST         0x0c    /* bus device reset                */
#define MSG_ID          0x80    /* identify message                */
#define MSG_IDD         0x80    /* identify with disconnect        */
/*#define MSG_IDD               0xc0    /* identify with disconnect     */

#define IO_TIMEOUT      0xf1    /* I/O command timeout             */

/***** Disk I/O Error Codes *****/

#define EC_NORMAL       0x00    /* normal command completion       */
#define EC_BAD_CMD      0x01    /* bad command from caller         */
#define EC_ADDR_MK      0x02    /* address mark not found          */
#define EC_REC_NF       0x04    /* record not found                */
#define EC_BAD_RST      0x05    /* reset failed                    */
#define EC_DMA_BND      0x09    /* attempt to DMA over 64K bound*/
#define EC_BAD_TRK      0x0b    /* bad track                       */
#define EC_BAD_ECC      0x10    /* ECC error on read               */
#define EC_DAT_REC      0x11    /* corrected data error            */
#define EC_CNTLR        0x20    /* controller error                */
#define EC_SEEK         0x40    /* seek error                      */
#define EC_TIMEOUT      0x80    /* timeout error                   */
#define EC_PARITY       0x81    /* parity error                    */
#define EC_CHK_CND      0x82    /* check condition                 */
#define EC_BUSY         0x83    /* target busy                     */
#define EC_BUS_RST      0x84    /* bus reset occurred              */
#define EC_SCSI         0x85    /* SCSI interface error            */
#define EC_ADAPTER      0x86    /* host adapter error              */
#define EC_SEL_TO       0x87    /* selection timeout               */

/********************  Disk / Tape Error Codes  ********************/

/* Class 0 Sense Codes */

#define ED_NOSENSE      0x00            /* no sense                 */
#define ED_NOINDEX      0x01            /* no index signal          */
#define ED_NOSEEK       0x02            /* no seek complete         */
#define ED_WFAULT       0x03            /* write fault              */
#define ED_NOTREADY     0x04            /* drive not ready          */
#define ED_NOTSELECTED  0x05            /* drive not selected       */
#define ED_NOTRK0       0x06            /* no track zero            */
#define ED_MULTDSEL     0x07            /* multiple drives selected */
#define ED_NOTLOADED    0x09            /* media not loaded         */
#define ED_INSUFCAP     0x0a            /* insufficient capacity    */
#define ED_DTO          0x0b            /* tape - drive timeout     */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
/* Class 1 Sense Codes */

#define ED_IDREADERR     0x10          /* ID read error                    */
#define ED_UDATAERR      0x11          /* Uncorrectable data error         */
#define ED_IDADDR        0x12          /* ID address mark not found        */
#define ED_DATAADDR      0x13          /* data address mark not found      */
#define ED_NOBLOCK       0x14          /* block not found                  */
#define ED_SEEKERR       0x15          /* seek error                       */
#define ED_DMATO         0x16          /* DMA timeout error                */
#define ED_WRTPROT       0x17          /* write protect                    */
#define ED_CDATACHK      0x18          /* correctable data check           */
#define ED_BADBLK        0x19          /* bad block found                  */
#define ED_INTERR        0x1a          /* interleave error                 */
/*#define ED_FM_DETECT   0x1c          /* tape - file mark detected        */
#define ED_CMDLATE       0x1c          /* command late error               */
#define ED_DATLATE       0x1d          /* data late error                  */
#define ED_WCHKERR       0x1e          /* write check error                */
#define ED_ECC           0x1f          /* ECC soft error                   */

/* Class 2 Sense Codes */

#define ED_INVCMD        0x20          /* invalid command                  */
#define ED_ILLBLKADDR    0x21          /* illegal block address            */
#define ED_UNITATT       0x30          /* unit attention                   */
#define ED_CMDTO         0x31          /* command timeout                  */

/* Extended Sense Error Codes */

#define EX_NO_SENSE      0x00          /* no sense key information         */
#define EX_REC_ERROR     0x01          /* recoverable error                */
#define EX_NOT_READY     0x02          /* LUN not ready                    */
#define EX_MEDIA_ERROR   0x03          /* media error                      */
#define EX_HARDWARE_ERR  0x04          /* hardware error                   */
#define EX_ILLGAL_REQ    0x05          /* illegal request                  */
#define EX_UNIT_ATT      0x06          /* unit attention                   */
#define EX_DATA_PROTECT  0x07          /* data protect                     */
#define EX_1RES          0x08          /* reserved                         */
#define EX_VENDOR_UNIQ   0x09          /* vendor unique                    */
#define EX_COPY_ABORTED  0x0a          /* copy aborted                     */
#define EX_ABORTED_CMD   0x0b          /* aborted command                  */
#define EX_2RES          0x0c          /* reserved                         */
#define EX_VOL_OVFL      0x0d          /* volume overflow                  */
#define EX_MISCOMPARE    0x0e          /* miscompare on verify             */
#define EX_3RES          0x0f          /* reserved                         */

/*****  Driver Structures  *****/

#ifdef  SUN
struct  short_blk {             /* 6-byte command block                 */
        UCHAR   lun_lba2;       /* LUN + MSB logical block addr         */
        UCHAR   cmd;            /* command group/code                   */
        UCHAR   lba0;           /* MSB logical block address            */
        UCHAR   lba1;           /* logical block address                */
        UCHAR   flag_link;      /* flag + link fields                   */
        UCHAR   num_blks;       /* number of blocks                     */
};
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
struct  long_blk {              /* 10-byte command block        */
        UCHAR   lun;            /* logical unit number          */
        UCHAR   cmd;            /* command group/code           */
        int     blk_addr;       /* logical block address        */
        UCHAR   msb_num_blks;   /* MSB number of blocks         */
        UCHAR   no_used;        /* not used                     */
        UCHAR   flag_link;      /* flag + link fields           */
        UCHAR   lsb_num_blks;   /* LSB number of blocks         */
};
#else
struct  short_blk {             /* 6-byte command block         */
        UCHAR   cmd;            /* command group/code           */
        UCHAR   lun_lba2;       /* LUN + MSB logical block addr */
        UCHAR   lba1;           /* logical block address        */
        UCHAR   lba0;           /* MSB logical block address    */
        UCHAR   num_blks;       /* number of blocks             */
        UCHAR   flag_link;      /* flag + link fields           */
};

struct  long_blk {              /* 10-byte command block        */
        UCHAR   cmd;            /* command group/code           */
        UCHAR   lun;            /* logical unit number          */
        int     blk_addr;       /* logical block address        */
        UCHAR   no_used;        /* not used                     */
        UCHAR   msb_num_blks;   /* MSB number of blocks         */
        UCHAR   lsb_num_blks;   /* LSB number of blocks         */
        UCHAR   lflag_link;     /* flag + link fields           */
};
#endif

struct  siocb {                 /* 6 & 10-byte command blk share*/
        union {                 /* same allocation storage      */
                struct short_blk sb;    /* short block          */
                struct long_blk  lb;    /* long  block          */
                short  ssbuf[MAX_IOCB_SZ];/* treat as short buf         */
        } c_un;
};

struct  dk_inf {
        int     dk_present;
        int     dk_blk_size;
};

struct  size {                  /* disk partitions              */
        long    nblocks;        /* blocks per partition         */
        long    blockoff;       /* block offset                 */
} ha_sizes[CNTLR_UNITS][DISK_UNITS][FS] = {
        S_BLKS_ROOT, 0,         /* root - file system 0         */
        S_BLKS_SWAP, 0,         /* swap file                    */
        S_BLKS_USR,  0,         /* usr  - file system 1         */
        S_BLKS_FAST, 0,         /* disk/tape disk buffer        */
        S_BLKS_BOOT, 0          /* bootstrap blocks             */
};

struct sioctl {
        struct  siocb iocb;     /* command block                */
#ifdef  SUN
        UCHAR   status;         /* status byte                  */
        UCHAR   msg;            /* message byte                 */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#else
        UCHAR   mss;            /* message byte                 */
        UCHAR   status;         /* status byte                  */
#endif
        int     bcount;         /* byte size of transfer        */
        char    *buf_addr;      /* data buffer pointer          */
};

struct  s_capacity {
        long    last_blk_addr;
        long    block_size;
};

struct  s_inquiry {
        UCHAR   dev_typ;
        UCHAR   dev_typ_qualifier;
        UCHAR   rev_level;
        UCHAR   alt_cyls;
};

struct s_sense {
        UCHAR   class_code;
        UCHAR   msb_blk_addr;
        UCHAR   sblk_addr;
        UCHAR   lsb_blk_addr;
};

struct x_sense {                        /* extended sense info  */
        UCHAR   vadd;
        UCHAR   ses_no;
        UCHAR   sense_key;
        UCHAR   bmsb_2;
        UCHAR   bmsb_1;
        UCHAR   blsb_2;
        UCHAR   blsb_1;
        UCHAR   addl_lsth;
        UCHAR   ercl_ercd;
        UCHAR   rec_msb_errs;
        UCHAR   rec_lsb_errs;
};

struct s_msense {                       /* for mode sense       */
        UCHAR   msize;                  /* size - 0x0c          */
        UCHAR   wp_mdty;                /* write prot&mdedia typ*/
        UCHAR   bufm_spd;               /* buffered mode, spd   */
        UCHAR   const8;                 /* always 0x08          */
        UCHAR   density;                /* density code         */
        UCHAR   num_1blks;              /* total blks on tape   */
        UCHAR   num_2blks;              /* total blks on tape   */
        UCHAR   num_3blks;              /* total blks on tape   */
        int     blk_size;               /* always 512           */
};
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
struct  refs {
        union {
                char    cbuf[SZ_CHARS];
                short   sbuf[SZ_SHORTS];
                long    ibuf[SZ_INTS];
                struct  s_capacity ss_capacity;
                struct  s_inquiry ss_inquiry;
                struct  s_sense ss_sense;
                struct  s_msense ss_msense;
                struct  x_sense ext_sense;
        } r_un;
};

struct timer_info {
        UCHAR   ha_tmo;
        UCHAR   ha_tflag;
} ha_tinfo;

/* Typedefs */

typedef struct sioctl sioctl_t;
typedef struct siocb siocb_t;
typedef struct refs refs_t;

/****** Driver Global variables *****/

siocb_t *iocb;                          /* The command block    */

sioctl_t *iiocb;                        /* The ioctl command block      */

refs_t  *data_refs;                     /* char,short,int references    */

siocb_t *iocb;                          /* The command block    */

sioctl_t *iiocb;                        /* The ioctl command block      */

refs_t  *data_refs;                     /* char,short,int references    */

#ifndef SUN
siocb_t  m_iocb;
sioctl_t m_iiocb;
refs_t   m_data_refs;
#endif

char    ha_topenf[TAPE_UNITS];          /* open flags for tape  */

char    ha_tflags[TAPE_UNITS];          /* special tape flags   */

char    ha_tpres[TAPE_UNITS];           /* present tape units   */

char    ha_unit_types[HA_UNITS];        /* device types         */

char    ha_inited = INIT_NO;            /* initialized flag     */

char    timer_valid = 0;                /* interrupt timing flag*/

char    msgin_buf;                      /* message in cell      */

char    msgout_buf;                     /* message out cell     */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
char    sts_buf;                        /* status cell          */

char    state_flag;                     /* phase state          */

int     status;                         /* status word          */

short   sem_ioctl = SEM_FREE;           /* ioctl semaphore      */

UCHAR   ioctl_status;                   /* ioctl status byte    */

UCHAR   ioctl_msg;                      /* ioctl msg byte       */

#ifdef  DEBUG
char    ha_debug  = 0;                  /* debug only           */
#endif
char    ha_force  = 0;                  /* debug only           */

#ifndef STANDALONE
struct  buf     hatab;                  /* queue head           */

struct  buf     rhabuf;                 /* for raw I/O          */

struct  buf     habuf;                  /* for driver issued I/O*/
#endif

struct dk_inf dk_info[CNTLR_UNITS][DISK_UNITS];

caddr_t addr_save;                      /* split I/O address save*/
/*short dummyx [0x1fa]; /**/



/*
        JGB Software Services           J. G. Beteta    7/10/84

        SCSI disk/tape driver   -       ha.c  &  ha.h

        Developed for Emulex Corporation
*/

#include "../h/param.h"
#include "../h/systm.h"
#include "../h/buf.h"
#include "../h/dir.h"
#include "../h/user.h"
#ifdef  SUN
#include "../h/dma.h"
#else
#define DMA_ADDR(x)     (x)
#endif

#include "ha.h"

extern ha_strategy();
extern ha_timer();

haopen(dev,flag)
dev_t dev;
int flag;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
{
        refs_t *dp;
        int     i,lun,scsi_dev,sts;

        scsi_dev = (minor(dev) >= TAPE) ? (tscsidev(dev) | TAPE) :
                                          scsidev(dev);
        lun = scsilun(dev);
        if((ha_inited == INIT_DONE) &&
          !(dk_info[scsi_dev & MASK_UNIT][lun].dk_present)) {
                u.u_error = ENXIO;
                return;
        }
        if(minor(dev) >= TAPE) {
                /*lun = scsilun(dev); /**/
                /*scsi_dev = tscsidev(dev) | TAPE;/**/
                dp = data_refs;
                if(lun >= TAPE_UNITS || ha_topenf[lun]) {
                        u.u_error = ENXIO;
                        return;
                }
                if(ha_command(scsi_dev,lun,OC_READY,
                        SZ_REPLY,data_refs)) {
                        printf("HA: cartridge tape unit is not ready\n");
                        ha_command(scsi_dev,lun,OC_SENSE,SZ_REPLY,data_refs);
                        u.u_error = ENXIO;
                        return;
                }
                /*ha_command(scsi_dev,lun,OC_SENSE,SZ_REPLY,data_refs);/**/
                ha_command(scsi_dev,lun,OC_LOAD,SZ_REPLY,data_refs);
                ha_command(scsi_dev,lun,OC_REWIND,SZ_REPLY,data_refs);
                ha_topenf[lun] = TRUE;
                u.u_error = 0;
        }
}

haclose(dev,flag)
dev_t dev;
int flag;
{
        int scsi_dev,lun;

        if(minor(dev) >= TAPE) {
                lun = scsilun(dev);
                scsi_dev = tscsidev(dev) | TAPE;
                ha_topenf[lun] = FALSE;
                if(ha_tflags[lun] & TF_WRITTEN) {
                        ha_command(scsi_dev,lun,OC_FM_WRITE,SZ_REPLY,data_refs);
                        ha_tflags[lun] = 0;
                }
                ha_command(scsi_dev,lun,OC_REWIND,SZ_REPLY,data_refs);
                ha_command(scsi_dev,lun,OC_LOAD,SZ_REPLY,data_refs);
        }
}

hastrategy(bp)
struct buf *bp;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
{
        int xx = 0xcc11;
        struct buf *dp;
        int unit;
        int unsigned x, sz;
        long laddr;
        int i;
        int zz = 0xcc12;

        if(ha_inited != INIT_YES)
                hainit();                       /* determine confisuration      */
        if(bp->b_flags & B_PHYS) {
#ifdef  SUN
                mapalloc(bp);
#endif
        }
        unit = minor(bp->b_dev) & MASK_UNIT;    /* file system                  */
        sz = bp->b_resid = bp->b_bcount;
        if(minor(bp->b_dev) < TAPE) {           /* disk operation               */
                sz = ((sz+BMASK) >> BSHIFT) * dk_info[scsidev(bp->b_dev)]
                        [scsilun(bp->b_dev)].dk_blk_size;
                if((ha_inited == INIT_DONE) && ((bp->b_blkno+sz >
                        ha_sizes[scsidev(bp->b_dev)][scsilun(bp->b_dev)]
                                [unit].nblocks) ||
                        (ha_sizes[scsidev(bp->b_dev)][scsilun(bp->b_dev)]
                                [FS_USR].blockoff == 0))) {
                        bp->b_flags |= B_ERROR;
                        iodone(bp);
                        return;
                }
                bp->av_forw = NULL;
                bp->b_error = 0;
                x = SPL();
                dp = &hatab;
                if(dp->b_actf == NULL)
                        dp->b_actf = bp;
                else
                        dp->b_actl->av_forw = bp;
                dp->b_actl = bp;
                if(dp->b_active == NULL)
                        hastart();
        } else {                                /* its a tape operation         */
                /* add tape code here   */
                bp->av_forw = NULL;
                bp->b_flags |= B_TAPE;
                bp->b_error = 0;
                x = SPL();
                dp = &hatab;
                if(dp->b_actf == NULL)
                        dp->b_actf = bp;
                else
                        dp->b_actl->av_forw = bp;
                dp->b_actl = bp;
                if(dp->b_active == NULL)
                        hastart();
        }

        splx(x);
}
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
hastart()
{
        int xx = 0xdd11;
        struct buf *bp;
        siocb_t *siocb = iocb;
        int unit, sz;
        daddr_t bn;
        int scsi_dev;
        int     tape_op = FALSE;
        int     dk_mod;
        int zz = 0xdd12;

        if((bp = hatab.b_actf) == NULL) {       /* nothing to do        */
                return;
        }
        addr_save = 0;
        hatab.b_active++;                       /* activate it          */
        unit = minor(bp->b_dev) & MASK_UNIT;    /* file system          */
        sz = bp->b_bcount;
        dk_mod = dk_info[scsidev(bp->b_dev)][scsilun(bp->b_dev)].dk_blk_size;
        if(minor(bp->b_dev) < TAPE) {
                bn = bp->b_blkno*dk_mod;
                if(bp != &habuf)
                        bn=bn+ha_sizes[scsidev(bp->b_dev)][scsilun(bp->b_dev)][un
                scsi_dev = scsidev(bp->b_dev);
                sz = ((sz + BMASK) >> BSHIFT) * dk_mod;
        } else {
                tape_op = TRUE;
                scsi_dev = tscsidev(unit);
                bn = 0;
                sz = ((sz + BMASK) >> BSHIFT) * 2;
        }

        unit = scsilun(bp->b_dev);      /* driver number        */
        /* setup the iocb       */
        ha_clear_iocb(siocb);
        if(bp != &habuf)  {
                bp->b_cylin = (bp->b_flags & B_READ) ? OC_READ : OC_WRITE;
        }
        if(bn <= MAX_SB) {              /* build 6-byte  iocb   */
                siocb->c_un.sb.cmd = bp->b_cylin;
                if(tape_op && (bp->b_cylin == OC_READ ||
                               bp->b_cylin == OC_WRITE))
                        siocb->c_un.sb.lun_lba2 = 1;
                else
                        siocb->c_un.sb.lun_lba2 = 0 | (unit << SHIFT_LUN)
                                        | ((bn >> 16) & 0x1f);
                siocb->c_un.sb.lba1 = ((bn >> 8) & MASK_BYTE);
                siocb->c_un.sb.lba0 = (bn & MASK_BYTE);
                siocb->c_un.sb.num_blks = sz;
                if(tape_op && (bp->b_cylin == OC_WRITE))
                        ha_tflags[unit] = TF_WRITTEN;
        } else {                                /* build 10-byte iocb   */
                siocb->c_un.lb.cmd = bp->b_cylin | (CMD_GRP_1);
                siocb->c_un.lb.lun = unit;
#ifdef  SUN
                SWAP_BINT(bn);
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#endif
                siocb->c_un.lb.blk_addr = bn;
                siocb->c_un.lb.msb_num_blks = (sz >> 8) & MASK_BYTE;
                siocb->c_un.lb.lsb_num_blks = sz & MASK_BYTE;
        }

        if(bp == &habuf)
                ha_special_cmds(tape_op,bp->b_cylin);
        ha_send_scsi_cmd(bp,siocb,scsi_dev);

}

#ifdef  SUN
haintr()
#else
haintr(level)
#endif
{
        struct buf *bp;
        siocb_t *siocb;
#ifdef  SUN
        int level = INT_NORM_LEVEL;
#endif
        int sbc;
        int int_flag;
        int buf_addr;
        int i,msg,s_flag,gsr_save;
        int error = 0;
        char c;

/*
if(ha_debug == 1) {
printf("haintr: level = %x >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n",level);
get_c();
}
*/
#ifdef  SUN
        if((ha_tinfo.ha_tflag == TIMER_RUNNING) && (ha_tinfo.ha_tmo == 0))
                level = INT_TMO_LEVEL;
#endif
        ha_tinfo.ha_tmo = 0;
        bp = hatab.b_actf;
        if(bp == (struct buf *)NULL)
                siocb = (siocb_t *)bp->b_resid;
        else
                if(bp->b_flags & B_IOCTL)
                        siocb = (siocb_t *)iiocb;
                else
                        siocb = iocb;
        gsr_save = int_flag = inb(GSR);
        if(int_flag & 1) printf("I/O DMA timeout\n");
        if(level == INT_NORM_LEVEL) {
                if(int_flag & GSR_RST) {
                        ha_wtrue_loop_on(GSR,GSR_RST);
                        status = EC_BUS_RST;
                        ha_set_state_flag(SF_ABT,TRUE);
                        if(ha_inited == INIT_YES)
                                ha_delay(INIT_DELAY);
                        for(sbc=0;sbc<1000;sbc++)
                                if(inb(DSR) == DSR_DC)
                                        break;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        if(inb(DSR) != DSR_DC)
                status = EC_ADAPTER;
        outb(CTR,CTR_PAR | CTR_RES); /* enable parity & reselect
        int_flag = INR_RESET;
} else {
        sbc = inb(ASR);
                if(sbc & ASR_PE) {
                        if((sbc  == ASR_PE) ||
                           ((ha_set_state_flag()) & SF_IO)) {
                                status = EC_PARITY;
                                ha_set_state_flags(SF_ABT,TRUE);
                                msgout_buf = MSG_ABT;
                        }
                }
                int_flag = inb(INR);
}
switch (int_flag) {
  case INR_BS:
        if(!(gsr_save & 4))
                ha_wfalse_loop_on(GSR,GSR_REQ);
        s_flag = ha_set_state_flag();/* phony int ? */
        if((s_flag & SF_CMP) || (s_flag & SF_DCN)) {
                outb(CMR,AC_TP | CMR_BYT);
                return; /* wait for disconnect  */
        }
        sbc =  (sbc >> SHIFT_PHASES) & MASK_PHASES;
        switch (sbc) {
          case PH_OUT_DATA:
          case PH_IN_DATA:
                ha_data_phase(sbc,bp->b_un.b_addr,bp->b_bcount);
                ARM_TIMER(TMO_INT);
                return;
          case PH_COMMAND:
                ha_cmd_phase(bp->b_resid);
                ARM_TIMER(TMO_INT);
                return;
          case PH_STATUS:
                sts_buf = ha_status_phase();
                if(bp->b_flags & B_IOCTL) /* save status */
                        ioctl_status = sts_buf;
                ARM_TIMER(TMO_INT);
                return;
          case PH_IN_MSG:
                msgin_buf = msg = ha_inmsg_phase();
                if(bp->b_flags & B_IOCTL)
                        ioctl_msg = msg;
                ha_set_state_flag(SF_CMP,TRUE);
                ARM_TIMER(TMO_INT);
                return;
          case PH_OUT_MSG:
                ha_outmsg_phase(msgout_buf);
                ARM_TIMER(TMO_INT);
                return;
          case PH_BAD1:
          case PH_BAD2:
                ha_bad_phase(sbc);
                break;
        }       /* end sbc case */
        break;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
                case INR_DCN:
                        s_flag = ha_set_state_flag();
                        if(s_flag & SF_DCN) {
                                ARM_TIMER(TMO_INT);
                                return;         /* wait for interrupt   */
                        } else
                                if(s_flag & SF_SEL) {
                                        status = EC_SEL_TO;/**/
                                }
                        ha_set_state_flag(~SF_CMP,FALSE);
                        break;
                case INR_FC:  /* Function Complete Interrupt  */
                        if((ha_set_state_flag()) & SF_MSGI)
                                ha_eval_mssin((bp->b_dev >> SHIFT_UNIT)
                                        & MASK_UNIT);
                        ARM_TIMER(TMO_INT);
                        return;
                case INR_RES: /* Reselection Interrupt        */
                        i = ha_state_flag(~SF_DCN,FALSE);
                        ARM_TIMER(TMO_INT);
                        return;
                case INR_IVC: /* Invalid Command Interrupt    */
                        ha_reset();
                        status = EC_ADAPTER;
                        break;
                case INR_RESET: /* Reset Interrupt            */
                        ARM_TIMER(TMO_INT);
                        return;
        }
} else
        if(level == INT_TMO_LEVEL) { /* driver timeout call     */
                sts_buf = IO_TIMEOUT;
                printf("HA: command timeout\n");
        }


if((siocb->c_un.sb.cmd == OC_SENSE) && (bp == NULL)) {
        return; /* reading sense info directly - not thru bufs */
}
if(addr_save != 0) {                    /* fragmented I/O       */
        bp->b_un.b_addr = addr_save;
        addr_save = 0;
}
if((sts_buf != 0) || (status != 0)) {
        bp->b_flags |= B_ERROR;         /* flag error condition */
        if((bp->b_error++ < MAX_ERRORS) &&
           (!(bp->b_flags & B_IOCTL))) {
                hastart();      /* restart the failing I/O      */
                return;
        }
}
if(((bp->b_error >= MAX_ERRORS) && (ha_inited == INIT_DONE) &&
   (!(bp->b_flags & B_IOCTL)))) {
        error = TRUE;
}
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        hatab.b_errcnt = 0;
        hatab.b_active = NULL;
        hatab.b_actf   = bp->av_forw; /* dequeue completed request      */
        if(!(bp->b_flags  & B_ERROR) || /* no error or recovered        */
           ((bp->b_flags  & B_ERROR) && (bp->b_error < MAX_ERRORS))) {
                if(bp->b_flags & B_ERROR)
                        bp->b_flags &= ~B_ERROR; /* recovered from error*/
                bp->b_resid = 0;
                bp->b_error = 0;
        } else {
                bp->b_resid = bp->b_bcount;
                bp->b_error = sts_buf;
                bp->b_flags |= B_ERROR;
        }
        if(bp->b_flags & B_ERROR) {
                if(ha_err_handler(bp->b_dev,sts_buf,status))
                        deverror(bp,siocb->c_un.sb.cmd,sts_buf & MASK_BYTE);
                else { /* not really an error */
                        bp->b_resid = 0;
                        bp->b_error = 0;
                        bp->b_flags &= ~B_ERROR;
                }
                ha_reset();     /**/
        }
        iodone(bp);
        hastart();
}

hainit()
{
        int     xx = 0xaa11;
        refs_t *dp;
        int     i,x,unit,lun,sts;
        char    c;
        int     zz = 0xaa12;

        if(ha_inited == INIT_DONE) return;
printf("Enter '1' to enable debug mode ");
c = getchar();
if(c == 'a') { ha_inited = INIT_YES; return;}
if(c != '\n') ha_debug = c & 0x0f;
        ha_inited = INIT_YES;
        status = mssout_buf = mssin_buf = 0;
#ifdef  SUN
        if(((iocb  = (siocb_t *)multimem(sizeof(siocb_t))) == NULL) ||
           ((iiocb = (sioctl_t *)multimem(sizeof(sioctl_t))) == NULL) ||
           ((data_refs = (refs_t *)multimem(sizeof(refs_t))) == NULL)) {
                printf("HA: Unable to allocate MULTIBUS memory\n");
                return;
        }
#else
        iocb = &m_iocb;
        iiocb = &m_iiocb;
        data_refs = &m_data_refs;
#endif
        /* initialize the bloody INTEL 8237 DMA chip    */
        dp = data_refs;
        outb(DMA_MC,0);         /* master clear the DMA channel         */
        outb(DMA_CMD,D_P1);     /* set extended write/normal timing     */
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        x = SPL();
        ha_reset();
        splx(x);
        ha_delay(100);

        for(unit=0;unit<2;unit++) {     /* for test only */
                for(lun=0;lun<1;lun++) { /* for test only */
                        sts = ha_command(unit,lun,OC_READY,SZ_REPLY,data_refs);
                        if(sts == 0) {
ha_unit_types[unit] = ID_DISK; /* force it - KLUDGE */
                                dk_info[unit][lun].dk_present = 1;
                                if((sts == 0) && (unit < 1)) {
                                   if((ha_command(unit,lun,OC_CAPACITY,
                                        SZ_REPLY,data_refs)) == 0) {
                                        ha_tbl_init(unit,lun,
                                SWAP_BINT(dp->r_un.ss_capacity.last_blk_addr),
                                 SWAP_BINT(dp->r_un.ss_capacity.block_size));
                                   }
                                }
                        }
                }
        }
        ha_inited = INIT_DONE;
        u.u_error = 0;
        ha_tinfo.ha_tmo = TIMER_RESET;
        timeout(ha_timer,0,TMO_INT);    /* arm interrupt timer  */
        ha_tinfo.ha_tflag = TIMER_RUNNING;
        ha_tinfo.ha_tmo = TMO_INT;
}

haread(dev)
dev_t dev;
{
#ifdef  SUN
        physio(hastrategy, &rhabuf, dev, B_READ);
#else
        physio(&hastrategy, &rhabuf, dev, B_READ);
#endif
}

hawrite(dev)
dev_t dev;
{
#ifdef  SUN
        physio(hastrategy, &rhabuf, dev, B_WRITE);
#else
        physio(&hastrategy, &rhabuf, dev, B_WRITE);
#endif
}

haioctl(dev,command,addr,flag)
dev_t dev;
int command;
caddr_t addr;
int flag;
{
        siocb_t  *siocb;
        sioctl_t *sioctl;
        int x;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        if(!suser()) {
                printf("HA: must be superuser\n");
                u.u_error = EPERM;
        }
        if(ha_inited != INIT_YES)
                hainit();                       /* determine confisuration      */
        if((command & MASK_BYTE) >= OC_ODEBUG) {
                ha_debus = command & MASK_DEBUG;
                return;
        }
        x = SPL();
        while(sem_ioctl)
                sleep((caddr_t)sem_ioctl,PRIBIO);
        sem_ioctl = SEM_BUSY;
        splx(x);
        sioctl = iiocb;
        ha_clear_iocb(sioctl);
        if(copyin(addr,sioctl,sizeof(sioctl_t))) {
                u.u_error = EFAULT;
        }
        if(u.u_error != 0) {
                sem_ioctl = SEM_FREE;
                wakeup((caddr_t)sem_ioctl);
                return;
        }
        /* need to lock user process in memory by calling physio */
        u.u_count  = sioctl->bcount;
        u.u_offset = 0;
        u.u_base = (caddr_t)sioctl->buf_addr;
        rhabuf.b_resid = (unsigned) addr;
#ifdef  SUN
        physio(ha_strategy,&rhabuf,dev,B_READ);
#else
        physio(&ha_strategy,&rhabuf,dev,B_READ);
#endif
        if(sem_ioctl) {
                sem_ioctl = SEM_FREE;
                wakeup((caddr_t)sem_ioctl);
        }
        /* this call places the status & msg at &sioctl+MAX_BYTES_IOCB */
#ifdef  SUN
        sushort(addr+MAX_BYTES_IOCB,(ioctl_msg |
                                (ioctl_status << SHIFT_BYTE)));
#else
        suword(addr+MAX_BYTES_IOCB,(ioctl_msg |
                                (ioctl_status << SHIFT_BYTE)));
#endif
}

ha_tbl_init(unit,lun,blk_max,blk_size)
int  unit,lun;
long blk_max,blk_size;
{
        int     i,dk_sz,dk_mod,dk_blk_size;
        long    sz,dk_capacity;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        dk_blk_size = (int)blk_size;
        /*dk_mod = (dk_blk_size == S512) ? SZ512 : SZ1024;/**/
        dk_mod = BSIZE / dk_blk_size;
        dk_info[unit][lun].dk_blk_size = dk_mod;
        if(dk_mod == SZ512)
                dk_capacity = (blk_max * blk_size)/(long)dk_mod;
        else
                dk_capacity = (blk_max * blk_size);
        if(dk_capacity < (SZ_SMALL_DK/(long)dk_mod))
                dk_sz = DK_SMALL;
        else
                if((dk_capacity >= (SZ_SMALL_DK/(long)dk_mod) &&
                   (dk_capacity <= SZ_MEDIUM_DK)/(long)dk_mod))
                        dk_sz = DK_MEDIUM;
                else
                        dk_sz = DK_LARGE;
        ha_sizes[unit][lun][FS_USR].nblocks = 0;
        switch (dk_sz) {
          case DK_SMALL:
                ha_sizes[unit][lun][FS_ROOT].nblocks = S_BLKS_ROOT/dk_mod;
                ha_sizes[unit][lun][FS_SWAP].nblocks = S_BLKS_SWAP/dk_mod;
                ha_sizes[unit][lun][FS_FAST].nblocks = S_BLKS_FAST/dk_mod;
                ha_sizes[unit][lun][FS_BOOT].nblocks = S_BLKS_BOOT/dk_mod;
                break;
          case DK_MEDIUM:
                ha_sizes[unit][lun][FS_ROOT].nblocks = M_BLKS_ROOT/dk_mod;
                ha_sizes[unit][lun][FS_SWAP].nblocks = M_BLKS_SWAP/dk_mod;
                ha_sizes[unit][lun][FS_FAST].nblocks = M_BLKS_FAST/dk_mod;
                ha_sizes[unit][lun][FS_BOOT].nblocks = M_BLKS_BOOT/dk_mod;
                break;
          case DK_LARGE:
                ha_sizes[unit][lun][FS_ROOT].nblocks = L_BLKS_ROOT/dk_mod;
                ha_sizes[unit][lun][FS_SWAP].nblocks = L_BLKS_SWAP/dk_mod;
                ha_sizes[unit][lun][FS_FAST].nblocks = L_BLKS_FAST/dk_mod;
                ha_sizes[unit][lun][FS_BOOT].nblocks = L_BLKS_BOOT/dk_mod;
                break;
        }

        sz = 0;
        for(i=1;i<FS+1;i++)
                sz += ha_sizes[unit][lun][i-1].nblocks;
        if(dk_mod == SZ512)
                blk_max /= 0x0002L;
        ha_sizes[unit][lun][FS_USR].nblocks = blk_max - sz;

        for(i=1;i<FS+1;i++) {
                ha_sizes[unit][lun][i].blockoff =
                   ha_sizes[unit][lun][i-1].nblocks  +
                   ha_sizes[unit][lun][i-1].blockoff;
        }
}

ha_timer()
{
        int     s;

        if(ha_tinfo.ha_tflag == TIMER_RUNNING) {
                if(ha_tinfo.ha_tmo && --ha_tinfo.ha_tmo == 0)  {
                        printf("HA:command time out\n");
                        s = SPL();
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#ifdef  SUN
                        haintr();
#else
                        haintr(INT_TMO_LEVEL);
#endif
                        splx(s);
                }
                timeout(ha_timer,0,TMO_INT);    /* arm interrupt timer  */
        }
}

ha_strategy(bp)                 /* special for ioctl commands           */
struct buf *bp;
{
        siocb_t *siocb;
        caddr_t addr;
        struct buf *dp;
        int scsi_dev;
        int x;

        addr = (caddr_t)bp->b_resid;
        siocb = (siocb_t *)iiocb;
#ifdef  SUN
        mapalloc(bp);
#endif
        if(minor(bp->b_dev) >= TAPE) {
                bp->b_flags |= B_TAPE;
                scsi_dev = tscsidev(bp->b_dev);
        } else
                scsi_dev = scsidev(bp->b_dev);
        bp->av_forw = NULL;
        x = SPL();
        dp = &hatab;
        if(dp->b_actf == NULL)
                dp->b_actf = bp;
        else
                dp->b_actl->av_forw = bp;

        dp->b_actl = bp;
        bp->b_resid = (unsigned)iiocb;
        bp->b_flags |= B_IOCTL;
        if(dp->b_active == NULL) {
                addr_save = 0;
                hatab.b_active++;
                ha_send_scsi_cmd(bp,siocb,scsi_dev);
        }

        splx(x);
}

ha_command(scsi_dev,lun,command,byte_count,dp)
int scsi_dev,lun,command,byte_count;
caddr_t dp;
{
        int xx = 0xbb11;
        struct buf *bp = &habuf;
        int s;
        int zz = 0xbb12;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        /*addr_save = 0; /**/
        s = SPL();
        while(bp->b_flags & B_BUSY) {
                bp->b_flags |= B_WANTED;
                sleep((caddr_t)bp, PRIBIO);
        }
        bp->b_flags = B_BUSY;
        splx(s);
        if(scsi_dev >= TAPE)
                bp->b_dev = 0 | (lun << SHIFT_UNIT) | (scsi_dev);
        else
                bp->b_dev = 0 | (lun << SHIFT_UNIT) | (scsi_dev << SHIFT_SCSI);
        bp->b_cylin = command;
        bp->b_bcount = byte_count;
        bp->b_blkno = 0;
        bp->b_un.b_addr = dp; /**/
        hastrategy(bp);
#ifdef  SUN
        iowait(bp);
#else
        if(ha_inited == INIT_DONE)
                iowait(bp);
        else
                my_iowait(bp);
#endif
        if(bp->b_flags & B_WANTED)
                wakeup((caddr_t)bp);
        bp->b_flags = 0;
        return(bp->b_error);
}

ha_send_scsi_cmd(bp,siocb,scsi_dev)
struct buf *bp;
siocb_t *siocb;
int scsi_dev;
{
        int     lun,i;
        char    c;

        if(siocb->c_un.sb.cmd & 0x20) {
          lun = siocb->c_un.lb.lun;
        } else  {
          lun = siocb->c_un.sb.lun_lba2 >> SHIFT_LUN;
        }

        if(bp == (struct buf *) NULL)
                bp->b_resid = (unsigned)iocb;
        else
                bp->b_resid = (unsigned)siocb;
        outb(DIR,scsi_dev);     /* set SCSI target device       */
        outb(TCR_B1,0x80);
        outb(TCR_B2,0x0f);
        outb(TCR_B3,0);
        ARM_TIMER(TMO_INT);
        status = 0;
/*if(!(ha_inited==INIT_DONE)) /**/
        outb(CMR,AC_SWA);       /* issue select with ATTN cmd   */
        msgout_buf = MSG_IDD | lun;
        ha_set_state_flag(SF_SEL,TRUE);
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
#ifndef SUN
/*ha_delay(1);/**/
#endif
        /* go away and wait on interrupt        */
}

ha_settimer()
{

        ha_tinfo.ha_tflag = TIMER_RUNNING;
        /*ha_tinfo.ha_tmo = TMO_INT; /**/
        timeout(ha_timer,0,TMO_INT);    /* arm interrupt timer  */
}

ha_dma(phase,buf_addr,byte_count,direction)
int phase,buf_addr,byte_count,direction;
{
        struct buf *bp;
        unsigned int nbuf_addr;
        unsigned int addr_iocb;
        int  es,b_count;
        long ax,bx;
        char a1,a2,a3;

        bp = hatab.b_actf;
        addr_iocb = (int)iocb;/* */
        /*addr_iocb = (int)&iocb; /* KLUDGE  to force an error */
        if((buf_addr == addr_iocb) ||
           (buf_addr == (int)data_refs) ||
           (buf_addr == (int)iiocb))
                nbuf_addr = DMA_ADDR(buf_addr);
        else
                nbuf_addr = (bp->b_flags & B_MAP) ? (unsigned)bp->b_un.b_addr :
                            DMA_ADDR(buf_addr);

#ifdef  SUN
        ax = nbuf_addr;
#else
        if((bp->b_flags & B_PHYS) && (phase != PH_COMMAND))
                ax = nbuf_addr;
        else
                ax = nbuf_addr + ADDR_CONV;
        if(buf_addr == (int)data_refs)
                ax = nbuf_addr + ADDR_CONV;
#endif
        a1 = ax & 0xff;
        a2 = (ax >> 8) & 0xff;
        a3 = (ax >> 16) & 0xff;
        byte_count--;
        if((((ax & 0xffffL)+byte_count) > 0xffffL) &&
             (bp->b_xmem == 0)) {
                b_count = (unsigned)0x10000L - (ax & 0xffffL);
                addr_save = bp->b_un.b_addr;
                bp->b_bcount = bp->b_bcount - b_count;
                bp->b_un.b_addr += b_count;
                byte_count = b_count - 1;
        }
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
        outb(DMA_MASK,DMA_CHNL_MASK);   /* set channel mask bit          */
        outb(DMA_CBPF,0);               /* clear byte ptr flip/flop      */
        outb(DMA_MODE,direction);       /* DMA direction: in or out      */
        outb(DMA_CNT,byte_count);       /* low byte of count             */
        outb(DMA_CNT,byte_count>>SHIFT_BYTE); /* high byte of count      */
        outb(DMA_ADR,a1);       /* low addr byte                  */
        outb(DMA_ADR,  a2); /* high addr byte           */
#ifdef  SUN
        outb(DMA_PR,a3); /* 24:8 of addr                */
#else
        if((bp->b_flags & B_PHYS) && (phase != PH_COMMAND))
                a3 = bp->b_xmem;
        outb(DMA_PR, a3); /* 24:8 of addr                */
#endif
        outb(DMA_MASK,DMA_SBT);         /* clear channel mask            */
        return(byte_count+1);
}

ha_data_phase(phase,buf_addr,byte_count)
int phase,buf_addr,byte_count;
{
        struct  buf *bp;
        siocb_t *siocb = iocb;
        refs_t  *dp = data_refs;
        int     i,k,mode,pio = 0;
        char    c;

        if(phase == PH_IN_DATA) {
                mode = DMA_IN;
                ha_set_state_flag(SF_IO | SF_DAT,TRUE);
        } else {
                mode = DMA_OUT;
                ha_set_state_flag(~SF_IO,FALSE);
                ha_set_state_flag(SF_DAT,TRUE);
        }
        bp = hatab.b_actf;
        if((siocb->c_un.sb.cmd == OC_SENSE) && (bp == NULL)) {
                byte_count = SZ_REPLY;
                buf_addr = (unsigned int)data_refs;
        }
        byte_count = ha_dma(phase,buf_addr,byte_count,mode);
        outb(TCR_B3,byte_count >> SHIFT_WORD);
        outb(TCR_B2,byte_count >> SHIFT_BYTE);
        outb(TCR_B1,byte_count);
        outb(CMR,AC_TI | CMR_DMA); /* issue transfer command */
}

ha_cmd_phase(siocb)
siocb_t *siocb;
{

        ha_dma(PH_COMMAND,siocb,MAX_BYTES_IOCB,DMA_OUT);
        outb(TCR_B1,MAX_BYTES_IOCB);
        outb(TCR_B2,0);
        outb(TCR_B3,0);
        outb(CMR,AC_TI | CMR_DMA); /* issue transfer command */
        ha_set_state_flag(~(SF_IO | SF_SEL),FALSE);
}
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
ha_status_phase()
{

        outb(CMR,AC_TI | CMR_BYT);
        ha_wfalse_loop_on(ASR,ASR_DAT);
        ha_set_state_flag(SF_IO,TRUE);
        return (inb(DAR));
}

ha_outmsg_phase(msg)
int msg;
{

        outb(CMR,AC_TI | CMR_BYT);      /* issue single byte trans */
        outb(DAR,msg);          /* send msg                     */
        ha_set_state_flag(~SF_IO,FALSE);
}

ha_inmsg_phase()
{

        outb(CMR,AC_TI | CMR_BYT);      /* issue single byte trans */
        ha_wfalse_loop_on(ASR,ASR_DAT);
        ha_set_state_flag(SF_MSGI | SF_IO,TRUE);
        return (inb(DAR));
}

ha_set_state_flag(mask,mode)
unsigned int mask;
int mode;
{
        struct buf *bp;
        int unsigned word;
        word = state_flag;
        if(mode)
                state_flag = word | mask;
        else
                state_flag = word & mask;
}

ha_set_state_flag()
{
        struct buf *bp;

        return(state_flag);
}

ha_eval_msgin(lun)
int lun;
{

        int     abort = FALSE;
        int     i;

        ha_set_state_flag(~SF_MSGI,FALSE);
        if(!(ha_set_state_flag()) & SF_ABT) {
                if(msgin_buf & BIT_UNIT) {      /* MSG_ID                   */
                        if((msgin_buf & MASK_UNIT) != lun) {
                                status = EC_SCSI; /* SCSI interface error */
                                abort = TRUE;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
                        }
                } else
                        switch (mssin_buf) {
                          case MSG_CMP:         /* command complete      */
                                status = EC_NORMAL;
                                ha_set_state_flag(SF_CMP,TRUE);
                                break;
                          case MSG_DCN:         /* disconnect            */
                                ha_set_state_flag(SF_DCN,TRUE);
                                break;
                          case MSG_SDP:         /* save data pointer     */
                                if(ha_set_state_flag() & SF_DAT)
                                  if((i=inb(TCR_B3)) != 0) {
                                        i = 0 | (inb(TCR_B2) << SHIFT_BYTE)
                                              | (inb(TCR_B1) & MASK_BYTE);

                                        /* need more code here */
                                   }
                                break;
                          case MSG_LCC:         /* linked command complete */
                          case MSG_LCF:         /* linked cmd cmplt & flag */
                                break;
                          case MSG_MPE:         /* message parity error */
                                status = EC_PARITY;
                                abort = TRUE;
                                break;
                          case MSG_REJ:         /* message rejected      */
                                status = EC_SCSI;
                                abort = TRUE;
                                break;
                          default:
                                printf("HA: invalid mssin code: %x\n",
                                        mssin_buf);
                        }
        }

        if(abort) {
                ha_set_state_flag(SF_ABT,TRUE);
                mssout_buf = MSG_ABT;
                outb(CMR,AC_ATN);       /* set attention         */
        }

        outb(CMR,AC_MA);                /* message accepted      */
}

ha_bad_phase(phase)
int phase;
{
}

ha_delay(delay)
int delay;
{
        long    i,j,k;

        k = (long)delay * 0x1000L;
        for(i=0;i<k;i++)
                j = j;
}
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
ha_clear_iocb(siocb)
siocb_t *siocb;
{
        int i;

        for(i=0;i<MAX_IOCB_SZ;i++)
                siocb->c_un.ssbuf[i] = 0;
}

ha_wfalse_loop_on(addr,mask)    /* waits until signal goes true */
int addr,mask;
{
        int c,i=0;

        while(!((c=inb(addr,mask)) & mask)) {
                if(i++ >= MAX_WAIT_TIME) {
                        if(ha_debug == 1) printf("HA: timeout\n");
                        /*if(ha_debug == 1) dbs(0);*/
                        break;
                }
        }
        /*return (c); /**/
}

ha_wtrue_loop_on(addr,mask)     /* waits until signal goes false */
int addr,mask;
{
        int c,i=0;

        while((c=inb(addr,mask)) & mask) {
                if(i++ >= MAX_WAIT_TIME) {
                        if(ha_debug == 1) printf("HA: timeout\n");
                        break;
                }
        }
        /*return(c);/**/
}

ha_reser()
{
        outb(GCR,GCR_RST);
        /*ha_delay(50);/**/
        outb(RSRR,0); /* */
        inb(GSR);
        inb(GSR);
}

ha_special_cmds(tape_op,cmd)
int tape_op,cmd;
{
        siocb_t *siocb = iocb;

        switch (cmd) {
          case OC_SENSE:
                siocb->c_un.sb.num_blks = SZ_REPLY;
                break;
          case OC_REZERO:
          /*case OC_REWIND:*/
          case OC_CAPACITY:
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
            case OC_READY:
                  siocb->c_un.sb.num_blks = 0;
                  break;
            case OC_FM_WRITE:
                  siocb->c_un.sb.num_blks = 1;    /* file marks to write  */
                  break;
            case OC_INQUIRY:
                  siocb->c_un.sb.num_blks = 10;
                  break;
            case OC_LOAD:
                  if(tape_op) {
                        if(ha_topenf[0])
                              siocb->c_un.sb.num_blks = RESET_LOAD_BIT;
                        else
                              siocb->c_un.sb.num_blks = SET_LOAD_BIT;
                  }
                  break;
          }
}

ha_err_handler(dev,sts,status)
int dev,sts,status;
{
        siocb_t *siocb = iocb;
        refs_t *dp;
        int s,scsi_dev,lun;
        int s_code,failure = TRUE;

        scsi_dev = (minor(dev) >= TAPE) ? tscsidev(dev) : scsidev(dev);
        lun      = scsilun(dev);
        if(status != 0) {
           switch (status)  {
           case EC_SEL_TO:
                if(ha_inited == INIT_DONE)
                      printf("HA: Selection Timeout\n");
                break;
           default:
                printf("HA: sts = %x\n",status);
                break;
           }
        }
        switch (sts) {
          case CS_PE:
                printf("Parity error\n");
                break;
          case CS_BSY:
                printf("Target is busy, SCSI device %d, lun %d\n",
                      scsi_dev,lun);
                break;
          case CS_ICS:
                printf("Intermediate completion status\n");
                break;
          case CS_REC:
          case CS_ERR:
                siocb->c_un.sb.cmd = OC_SENSE;
                siocb->c_un.sb.lun_lba2 = 0 | (lun << SHIFT_LUN);
                siocb->c_un.sb.lba1 = 0;
                siocb->c_un.sb.lba0 = 0;
                siocb->c_un.sb.num_blks = SZ_REPLY;
                dp = data_refs;
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

```
                ha_send_scsi_cmd(0,sioctb,scsi_dev);/**/
                ha_delay(1);
                s = spl0();
                ha_delay(5);
                dp = data_refs;
                splx(s);
                s_code = dp->r_un.ext_sense.sense_key;
                if(s_code <= EX_REC_ERROR) { /* recoverable error */
                        failure = FALSE;    /* not valid error  */
                } else
                        if(ha_inited == INIT_DONE) {
                                printf("HA:error code=%x,ERCL=%x,ERCD=%x\n",
                                 s_code & MASK_NIB,
                                 (dp->r_un.ext_sense.ercl_ercd >> 4),
                                 (dp->r_un.ext_sense.ercl_ercd & MASK_NIB));
                        }
                break;
        }
        return(failure);
}

set_c()
{
        return(setchar());      /**/
}

dbs()
{
#ifdef  SUN
        debug(0);       /**/
#endif
}

#ifndef SUN
multimem(addr)
int addr;
{
        return(addr);
}

sushort(w1,w2)
int w1,w2;
{}
#endif

my_iowait(bp)
struct buf *bp;
{
        int s,i = 0;

        while((bp->b_flags & B_DONE) == 0) {
                s = spl0();
                ha_delay(1);
                splx(s);
                if(i++ > 25) {
                        break;
                }
        }
        if(i > 25)
                haintr(INT_TMO_LEVEL);
}
```

Figure 2-1. Source Listing for Xenix Driver Example (Cont'd)

EMULEX

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publication.

Manual Part Number _____ **Rev.** _____

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

☐ Please send me the current copy of the *Controller Handbook*, which contains the information on the remainder of EMULEX's controller products.

Name _____ Street _____

Title _____ City _____

Company _____ State/Country _____

Department _____ Zip _____

Additional copies of this document are available from:

Emulex Corporation
3545 Harbor Boulevard
P.O. Box 6725
Costa Mesa, CA 92626
Attention: Customer Services

Fold Here and Staple

Do Not Tear — Fold Here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 202 COSTA MESA. CA

POSTAGE WILL BE PAID BY ADDRESSEE

EMULEX CORPORATION
3545 HARBOR BOULEVARD
P.O. BOX 6725
COSTA MESA, CALIFORNIA 92626
ATTN: TECHNICAL PUBLICATIONS